Simulation énergétique de tâches distribuées avec changements dynamiques de fréquence

Tom Guérout, Thierry Monteil, Georges Da Costa, Mihai Alexandru

ComPAS'2013 : RenPar'21 / SympA'15/ CFSE'9
Grenoble, France, du 16 au 18 janvier 2013

Simulation énergétique de tâches distribuées avec changements dynamiques de fréquence

Tom Guérout[1,2,3], Thierry Monteil[1,3], Georges Da Costa[1,2,3], Mihai Alexandru[1,3]

[1]CNRS ; LAAS ; 7 avenue du Colonel Roche, F-31077 Toulouse, France
[2]IRIT ; 118 route de Narbonne, 31000 Toulouse, France
[3]Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
tguerout@laas.fr

Résumé
Ces dernières années, de nombreuses recherches ont été menées dans le domaine de la simulation des systèmes distribués, afin d'analyser et de comprendre leur comportement. Certains de ces simulateurs se focalisent sur le problème d'ordonnancement de tâches, d'autres sont spécifiquement développés pour la modélisation du réseau et seulement peu d'entre eux proposent tous les outils nécessaires pour simuler la consommation énergétique d'une application, d'une machine ou d'un centre de calcul. Cet article décrit les outils qui doivent être intégrés dans un simulateur pour être en mesure de lancer des simulations destinées à améliorer le comportement énergétique des machines. L'accent est mis davantage sur le DVFS (Dynamic Voltage and Frequency Scaling) et sa mise en œuvre dans CloudSim, le simulateur qui a été utilisé pour les expériences décrites dans cet article, mais aussi sur la façon de simuler et la méthodologie adoptée pour assurer la qualité des mesures et des simulations.

Mots-clés : simulations, énergie, dvfs

1. Introduction

De nos jours, les simulateurs sont, en autre, utilisés pour étudier la consommation énergétique des centres de calcul. En effet, l'utilisation croissante de ces centres de calcul rend l'analyse de leur consommation d'énergie de plus en plus importante. Certaines métriques afin d'évaluer leur efficacité énergétique sont connues (PUE, ERE) [17] et beaucoup de recherches sont en cours pour trouver de nouvelles méthodes de réduction d'énergie. Cela engendre une amélioration et/ou la création de nouveaux algorithmes qui nécessitent des phases de tests approfondies afin de valider leur efficacité.
Chacune des phases de développement et de tests de ces algorithmes sont très longues en raison du nombre de tentatives nécessaires pour vérifier et améliorer leur performance. De réelles plates-formes peuvent être utilisées pour dérouler ces phases de validation, mais cela engendre un temps de préparation non négligeable et les mesures physiques ne sont pas toujours possibles ou faciles à faire en fonction de l'équipement disponible. C'est pourquoi, les simulateurs sont de plus en plus couramment utilisés dans ce domaine.
Certains de ces simulateurs possèdent déjà quelques outils concrets de réduction de la consommation des machines/cpu, d'autres donnent la possibilité de mesurer la consommation énergétique totale d'une simulation, mais il semble qu'aucun d'entre eux ne réunissent tous les outils nécessaires afin de fournir un simulateur complet. Un autre point important est de savoir comment utiliser ces simulateurs, c'est pourquoi la partie validation des expériences prend une part importante de cet article.
Cet article est organisé comme suit. Premièrement un état de l'art des modèles énergétiques, des outils de réduction de consommation d'énergie et des simulateurs compose la section 2. Ensuite, la section 3 décrit toutes les modifications qui ont été ajoutées dans CloudSim. La section 4 présente des explications sur l'expérimentation et la validation du simulateur, puis une comparaison entre la simulation et

l'exécution réelle d'une application en modélisation électromagnétique est faite en section 5. Enfin, la section 6 conclut et expose des idées sur des travaux futurs.

2. État de l'art

2.1. Modèle énergétique

Dans [18], Rivoire établit une liste et une comparaison entre beaucoup de modèles énergétiques différents. Deux catégories peuvent être définies : des modèles très précis et détaillés , et des modèles de haut niveau appelés *black-box models*.
Des modèles du CPU très fins ont été proposés par Joseph et Martonosi [10] et par Isci et Martonosi [9]. Ces modèles sont très précis du point de vue de la modélisation de la consommation des processeurs et s'appuient sur des détails spécifiques de la micro-architecture de certains modèles de CPU.
Les modèles dits de haut niveau sont plus généraux à tous points de vue, cela augmente leur portabilité et leur rapidité. Ces modèles simples sacrifient un peu la précision des calculs, mais n'obligent pas à avoir une connaissance très détaillée de l'architecture des processeurs utilisés.
Dans cet article, le modèle de puissance utilisé est le modèle linéaire (à fréquence constante) $P_{TOT} = (1-\alpha) * P_{CPUIdle} + \alpha * P_{CPUFull}$, avec α la charge CPU, $P_{CPUIdle}$ et $P_{CPUFull}$ les puissances délivrées par le CPU à 0% et 100% d'utilisation. Bien sûr, certains modèles plus précis pourraient être utilisés comme dans [6].

2.2. Méthodes de réduction d'énergie

La première solution, appelée méthode ON/OFF, permet d'éteindre les machines trop peu utilisées (par rapport à un seuil d'utilisation) et de pouvoir les rallumer seulement si cela est nécessaire. Le cas échéant tous les processus qui s'exécutent dessus sont déplacés sur d'autres machines, et ainsi, les machines précédemment sous-utilisées, peuvent être éteintes. À l'inverse, lorsque toutes les machines sont sur-utilisées et que la demande est trop forte, alors une ou plusieurs machines sont redémarrées.

Comme décrit juste au dessus, il est parfois nécessaire de déplacer des processus de machines en machines. C'est le mécanisme de migration [11]. Il permet de déplacer une machine virtuelle (et tout son environnement) d'une machine à une autre. Ces déplacements ne sont pas gratuits en terme d'énergie car, chaque mouvement demande un certain temps et il est donc également important de prendre en compte le coût total d'une telle action. Cette technique permet de libérer des machines et ensuite de les éteindre, et en même temps d'utiliser les machines en fonctionnement à leur potentiel maximum (consolidation).

Enfin le DVFS (Dynamic Voltage and Frequency Scaling) [13] permet de modifier dynamiquement la tension et la fréquence des CPUs d'une machine en fonction de leur taux d'utilisation. Dans le noyau Linux, le DVFS peut être activé dans 5 modes différents : Performance, PowerSave, UserSpace, Conservative et OnDemand. Chaque mode a un *gouverneur* qui décide si la fréquence doit être modifiée (augmenter ou diminuer). Le *gouverneur* fonctionne avec un seuil haut et un seuil bas et vérifie périodiquement si la charge CPU est inférieure (ou supérieure) à ces seuils pour prendre sa décision. Baisser la fréquence CPU réduit la consommation, cela ralentit une application consommant beaucoup de calcul, mais sans affecter le temps passé en E/S ou en communication.

2.3. Simulateurs

SIMGRID [5] est un simulateur qui fournit des fonctionnalités de base pour la simulation d'applications distribuées hétérogènes dans des environnements distribués. Son utilisation est parfaitement adaptée à l'évaluation des heuristiques, au prototypage, au développement ou à l'amélioration d'applications réelles plutôt dans le domaine des grilles.

GroudSim [16] propose des outils de simulation évènementiels pour les grilles de calcul et le cloud computing. Les principaux domaines de simulation sont les transferts de fichiers, la charge des ressources disponibles et le calcul des coûts de fonctionnement. Il peut être utilisé avec un package contenant des

informations sur la probabilité de défaillance matérielle, le module d'événements peut alors détecter des erreurs se produisant sur la machine ou dans le réseau et lancer une procédure de reconfiguration sur les entités concernées.

GSSIM [2] [14] est un simulateur qui permet l'étude des politiques d'ordonnancement par une description de l'architecture très flexible et des interactions entre les modules d'ordonnancement. Les formats standards de workflow sont acceptés, "Standard Workload Format» (SWF) et "Grid Workload Format» (GWF). Un fichier XML supplémentaire peut être défini pour inclure des informations plus détaillées (exemple : contraintes de temps). GSSIM permet la virtualisation et intègre aussi des modèles énergétiques ainsi qu'une représentation précise de l'utilisation des ressources.

GreenCloud [12] est une extension du simulateur de réseau Ns2 [8]. Il offre aux utilisateurs une modélisation et une analyse détaillées de l'énergie consommée par les éléments qui composent le réseau : serveurs, routeurs et les liens entre eux. De plus, il permet d'analyser la répartition de la charge dans le réseau, ainsi que les communications avec grande précision (paquets TCP). Sur le plan de l'énergie GreenCloud définit trois types de consommation d'énergie : calcul (CPU), communications et physique du centre de calcul (système de refroidissement) et intègre deux méthodes de réduction d'énergie : le DVS (Dynamic Voltage Scaling) et le DNS (Dynamic Network Shutdown).

CloudSim [4], est un simulateur de Cloud Computing basé sur le simulateur GridSim [3]. L'architecture du centre de calcul est très bien modélisée et peut être facilement modifiée par l'utilisateur en fonction de ses besoins. CloudSim intègre la virtualisation, permet le développement de politiques de placement, la simulation réseau inter-centre de calcul ainsi que les coûts d'un service IaaS (Infrastructure as a Service) de Cloud Computing. Coté énergie, CloudSim permet l'extinction de machine, la migration de machines virtuelles et l'intégration de modèles énergétiques.

Après référencement de tous ces simulateurs, il semble qu'aucun d'entre eux n'a le DVFS réellement implémenté. Notre choix s'est porté sur CloudSim car ce dernier réunit quasiment tous les critères pour créer des simulations permettant l'analyse de la consommation d'énergie d'une expérience.

3. Intégration du DVFS dans CloudSim

Dans un *package* DVFS, les *gouverneurs* des 5 modes du DVFS, tel qu'ils sont présents dans le noyau Linux comme décrits dans la section 2.2 ont été implémentés. Leur rôle est de déterminer si la fréquence du CPU doit être modifiée et leur décision est directement liée à leur règle de décision intrinsèque. Dans le simulateur un changement de fréquence impacte donc directement la capacité en MIPS (Millions Instructions Per Second) des CPUs.
L'utilisation du DVFS impacte directement la capacité d'exécution des CPU (et donc des machines) qui devient variable dans cette version de CloudSim. Ceci implique également de modifier la manière dont le simulateur gère le placement et la gestion des machines virtuelles. Par exemple si le système décide de réduire la fréquence, la somme des capacités de toutes les VM peut temporairement dépasser la capacité maximale de la machine qui les héberge. Dans ce cas, la taille de chaque VM doit être revue temporairement à la baisse proportionnellement à la nouvelle capacité de la machine. La même situation se produit lorsque que de nouvelles machines virtuelles doivent être hébergées. Si la somme des capacités de toutes les machines virtuelles en cours d'exécution et celles des nouvelles machines virtuelles créées dépassent la capacité de la machine qui les accueille, le même processus de réduction de capacité des machines virtuelles est appliqué avant d'accueillir celles-ci dans la machine.
Également, lorsqu'une partie de la capacité d'une machine est libérée, on peut regarder si la capacité des machines virtuelles toujours en fonctionnement peut être augmentée. Ce cas a lieu quand une machine virtuelle a terminé son exécution ou lorsque la fréquence du CPU a été augmentée, toutes les capacités des machines virtuelles sont alors augmentées proportionnellement à la capacité libre sur la machine, tout en veillant à ne pas dépasser la capacité maximale de départ.
L'ajout du DVFS dans CloudSim a également nécessité de modifier la gestion des événements. En effet, le mode PowerSave induit un certain retard dans l'exécution des tâches en raison du CPU qui fonctionne

à sa fréquence plus basse. Dans ces conditions, l'évolution des événements séquentiels doit être directement liée avec la fin de chaque événement, la description statique des événements ne tenant pas compte de ce retard d'exécution. Ainsi, une nouvelle option a été ajoutée pour créer des événements à la fin de l'exécution d'un *broker* : un *broker* peut contenir une ou un ensemble de tâches, appelées *cloudlet* dans CloudSim. Au moment de la création du *broker*, on peut lui spécifier un *post-event* ce qui déclenchera sans délai un nouvel évènement à la fin de celui-ci.
La dernière modification apportée au sein du simulateur concerne le modèle énergétique. En effet, le modèle énergétique utilisé dépend de la machine que l'on veut comparer avec le simulateur. La définition du modèle utilisé dans cette expérimentation est décrite dans la section 4.2.

4. Validation mono-machine

L'objectif de cette expérimentation est de valider le comportement du DVFS et le calcul de consommation d'énergie dans CloudSim. L'idée principale est d'exécuter une séquence d'instruction sur une machine réelle, ayant un noyau Linux avec le DVFS activé et de mesurer la consommation totale dépensée. Ensuite, il s'agit de simuler exactement la même expérimentation dans CloudSim puis, activer le DVFS et comparer les valeurs de consommation d'énergie entre CloudSim et celle de la machine réelle dans chaque mode de DVFS.

4.1. Cadre expérimental

Toutes les expérimentations ont été effectuées sur une machine standard (appelée HOST par la suite) équipée d'un processeur Intel (R) Core (TM) 2 Quad CPU Q6700@2.66GHz et 4 Go de Ram utilisée sous Ubuntu TLS 10.4 (noyau Linux 2.6.32). Toutes les mesures de consommation d'énergie ont été réalisées avec un "plogg" powermètre sans fil [15] qui permet de voir et d'enregistrer la consommation d'énergie en temps réel.

4.2. Étalonnage de la consommation d'énergie :

Afin de calibrer le calcul de la consommation d'énergie dans le simulateur, il faut tout d'abord connaître les valeurs de fréquence admises par le CPU de la machine. Puis les valeurs de puissance délivrées, à 0% et 100% d'utilisation CPU pour chaque fréquence (soit 2*Nb_Freq mesures). Dans CloudSim la fréquence s'exprime en MIPS (Million d'Instructions Par Seconde). Les différentes fréquences ont donc été calculées proportionnellement à celle de la machine HOST.

Fréquences					
HOST	2.67	2.40	2.133	1.867	1.60
%	100	89.89	79.89	69.93	59.925
CloudSim	2500	2247	1997	1748	1498

TABLE 1 – Fréquences utilisables sur la machine HOST (en GHz), et celles introduites dans CloudSim (en MIPS)

Charge	Fréquences (GHz)				
	1.60	1.867	2.113	2.40	2.67
0%	82.7	82.85	82.95	83.10	83.25
100%	88.77	92.00	95.5	99.45	103.0

TABLE 2 – Puissances délivrées par la machine HOST à un taux d'utilisation de 0% et 100% pour chaque fréquence

4.3. Méthodologie

Le défi consiste à exécuter une expérimentation qui implique de nombreux changements de fréquences afin de tester le comportement du DVFS, mais aussi des phases de charge constantes pour utiliser le modèle de puissance sur toute sa gamme.
Chronologiquement, le déroulement de l'expérience (visible sur la figure 2(a)) est : (A) une phase de montée en charge régulière de 0 % à 96 %, (B) une phase de pleine charge à 96 %, (C) une phase de redescente linéaire jusqu'à 50%, (D) un pic de charge à 80 %, (E) une phase de redescente linéaire jusqu'à 30 %, (F) un pic de charge à 96 %, (G) une phase de charge constante de 30 %.

Pour exécuter ce scénario sur la machine HOST, une application de test a été implémentée en langage C. La charge voulue est obtenue en exécutant une boucle dont chaque itération est composée d'une période de calcul et d'une période de sommeil. Lorsque le DVFS est activé, la charge CPU est vérifiée

tous les "*sampling_rate*" (10ms). Pour être sûr que la décision du *gouverneur* DVFS soit prise directement en relation avec la charge CPU générée par la boucle de calcul, chaque temps d'itération de cette boucle doit être exactement égale à l'intervalle "*sampling_rate*".

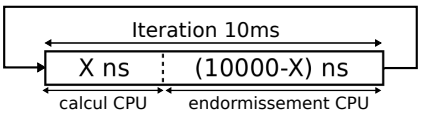
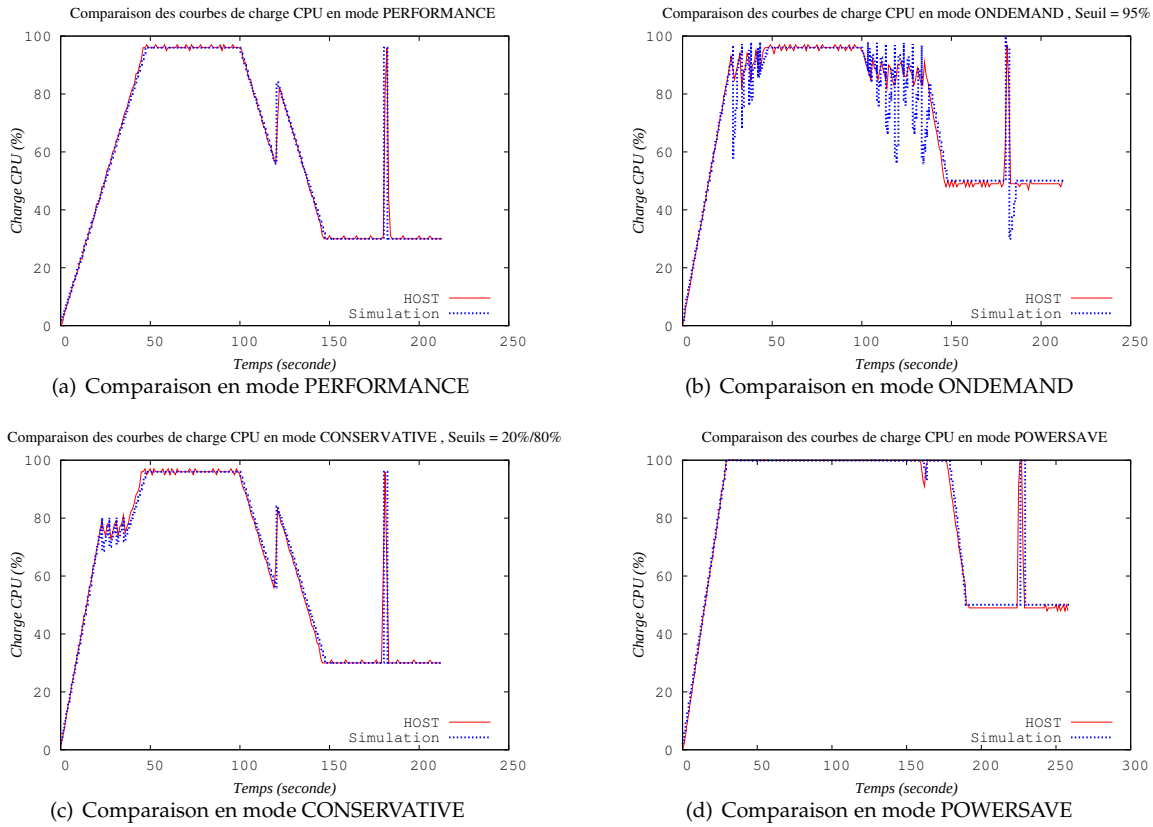


FIGURE 1 – Décomposition de la boucle en langage C

Ce scénario est créé dans CloudSim en exécutant un ensemble de *broker*, chacun contenant des VM dans lesquelles des *cloudlet* sont exécutés. Comme expliqué dans la section 3, un *broker* peut contenir un *post-event* qui est lancé à la fin de l'exécution du *broker*. En définissant différents types de VM (MIPS) et différents types de *cloudlet* (nombre d'instructions), il est possible d'augmenter et de gérer la charge du CPU avec précision. La taille d'une VM définit le pourcentage d'augmentation de la charge du processeur et la taille des *cloudlet* définit le temps d'exécution des VM. Donc, ce scénario est créé par un ensemble de *broker* qui eux mêmes lancent une série de couple VM/*cloudlet*. Enfin, pour respecter le même intervalle "sampling_rate" que dans le noyau Linux, l'intervalle de temps de simulation est aussi fixé à 10ms.

4.4. Graphique et Résultats

4.4.1. Comparaison de la charge CPU



(a) Comparaison en mode PERFORMANCE
(b) Comparaison en mode ONDEMAND
(c) Comparaison en mode CONSERVATIVE
(d) Comparaison en mode POWERSAVE

FIGURE 2 – Comparaison des charges CPU entre la machine réelle HOST et la simulation CloudSim dans les différents modes de DVFS

La figure 2(a) présente les courbes de charge CPU obtenues à fréquence maximale lors de l'expérience sur la machine réelle HOST et celle obtenues lors de la simulation dans CloudSim. A fréquence maxi-

male l'enjeu était d'avoir exactement la même charge CPU, pour être sûr que l'expérience et la simulation commencent dans les mêmes conditions. Ensuite, l'expérience à été lancée dans les 3 autres modes DVFS pour évaluer le comportement du simulateur.

4.4.2. Comparaison de la consommation énergétique

Le tableau 3 récapitule les temps d'exécution et les consommations d'énergie des différentes expériences et simulations, puis les compare en terme de taux d'erreur.

Mode DVFS	HOST		CloudSim		Erreur (%)
	Durée (s)	Consommation (Wh)	Durée (s)	Consommation (Wh)	
PERFORMANCE	213	5.72	213	5.61	-1.92
ONDEMAND	213	5.57	213	5.49	-1.43
CONSERVATIVE	213	5.68	213	5.64	-1.71
POWERSAVE	259	6.37	260	6.33	-0.63

TABLE 3 – Comparaison des résultats de consommation énergétique entre la machine HOST et CloudSim dans les quatre modes de DVFS

4.4.3. Analyse des résultats

Il est intéressant de constater que sur un exemple simple comme celui-ci, les résultats, au point de vue du temps d'exécution et des consommations énergétiques respectent la logique des différents modes du DVFS et qu'il y a une cohérence entre les mesures réelles et les résultats du simulateur. Les modes OnDemand (figure 2(b)) et Conservative (figure 2(c)), ont des temps d'exécution similaire au mode Performance car il n'implique pas (ou très peu) de ralentissement d'exécution en augmentant la fréquence des CPUs directement lorsque cela est nécessaire, et ainsi le mode OnDemand est celui qui donne le meilleur résultat de consommation énergétique car il est plus réactif lors des diminutions de fréquence que le mode Conservative. Le résultat du mode PowerSave est compréhensible, car même si théoriquement c'est ce mode qui donne les consommations les plus basses à un instant t, l'expérience est beaucoup plus longue, dûe à la faible fréquence du CPU, ce qui n'est pas favorable d'un point de vue énergétique.

5. Application Électromagnétique : TLM

Ayant analyser le comportement du DVFS dans CloudSim avec une application simple dans la section précédente, cette section présente d'autres résultats de comparaison avec une application parallèle plus complexe implémentant la méthode TLM (Transmission Line matrix Method).
La TLM est une méthode numérique pour la simulation électromagnétique qui représente un environnement de propagation de champs électromagnétique à travers un réseau de lignes de transmission. Ce modèle de propagation réside sur l'équivalence qui existe entre les champs électriques et magnétiques ainsi qu'entre les tensions et les courants dans un réseau de lignes de transmission. Une présentation détaillée de la méthode TLM, est décrite dans [7].
La résolution de cette méthode de manière parallèle sur une grille de calcul, est basée sur un découpage de la structure suivant les trois axes. Les volumes obtenus sont assimilés à des tâches, exécutées chacune sur une machine, un CPU, ou un coeur de processeur, échangeant des informations via la bibliothèque MPI (Message Passing Interface).

5.1. Solveur CPLEX

Le solveur IBM Ilog CPLEX a été utilisé pour déterminer la fréquence optimale fixe, en fonction des paramètres d'entrée de la TLM (taille de la structure et nombre d'itérations), du débit et de la latence réseau et des valeurs de puissance des machines. Le problème a été décrit comme suit :
H : Nœud utilisé
T_{net} et T_{cpu} : Temps réseau et Temps CPU.
$F = \{f_1, f_2, ..., f_{n-1}, f_n\}$: Fréquences disponibles sur le noeud H, avec $f_{n-1} < f_n$.
$l^H_{cpu}(f_i)$ et $d^H_{cpu}(f_i)$: Puissances délivrées par le noeud H à 0% et 100% d'utilisation CPU.
z^f_i : Variable binaire, égale à 1 si f_i est utilisée, 0 sinon.

Le modèle de prédiction de temps CPU et temps réseau de la TLM présenté dans [1] a été utilisé : $T_{cpu} = C_1 + n_l n_x n_y n_z C_2$, $T_{net} = \left(L + \frac{n_x n_y}{D}\right) * 4n_l$, avec n_x, n_y,n_z les dimensions de l'environnement TLM, n_l le nombre d'itérations, C_1 et C_2 des constantes estimées par des expérimentations passées, L et D la latence et le débit réseau.
L'expression de l'énergie E à minimiser est :

$$E = \quad T_{net} * \left[l^H_{cpu}(f_1) * z^f_a + ... + l^H_{cpu}(f_n) * z^f_n\right] + (T_{cpu} * f_n) \left[\frac{d^H_{cpu}(f_1) * z^f_a}{f_1} + ... + \frac{d^H_{cpu}(f_n) * z^f_n}{f_n}\right]$$

avec la contrainte : $\sum_{i=1}^{n} z^f_i = 1$, qui autorise l'utilisation d'une seule fréquence à la fois.

5.2. Expérimentations Grid'5000 et simulations CloudSim

Les expérimentations ont été menées sur le site de Reims de Grid'5000, dont la configuration est : HP Proliant DL165 G7, CPU :AMD Opteron 6164 HE 1.7GHz, 12 MB L3 cache, 44 nœuds de 2 CPUs avec 12 cœurs par CPU (1056 cœurs). La TLM a été exécutée sur deux nœuds utilisés à 100%, soit sur 48 cœurs, avec une tâche TLM sur chacun des cœurs. Les paramètres d'entrée utilisés sont : n_x=172, n_y=90, n_z=12, n_l=26000. Le débit et la latence mesurés sur Reims sont : D=700Mbit/s et L=4*10^{-5}s.

5.3. Résultats

Les valeurs en gras dans le tableau 4 sont les valeurs obtenues avec CloudSim, les autres sont les valeurs moyennes relevées lors des expérimentations sur Grid'5000.

Mode DVFS	**Durée (s)**			**Énergie (Wh)**		
	Moyenne	Ecart-Type	% Erreur	Moyenne	Ecart-Type	% Erreur
Performance (1.7GHz)	**8320** 8315	414	-0.06	**833** 832	75	-0.12
Optimale (1.5GHz)	**8580** 8262	367	-3.7	**831** 832	30	0.12
OnDemand	**8320** 8320	367	0	**743** 827	40	11.31

TABLE 4 – Comparaison des résultats de temps d'exécution et de consommation entre les expériences Grid'5000 (moyennes calculées sur environ 5 expériences) et CloudSim

5.3.1. Analyse des résultats

Concernant les résultats obtenus avec le simulateur, on peut remarquer que l'on retrouve la logique attendue : la fréquence optimale théorique donnée par le solveur obtient un résultat de consommation plus faible que la fréquence maximale, malgré un temps d'exécution plus long et le mode OnDemand est encore plus économique avec un gain -10.5% par rapport à la fréquence optimale. Les soucis de latence réseau et de débit sont gommés dans le simulateur, alors que les valeurs relevées lors des expérimentations réelles sont très influencées par ces variations. En analysant les résultats de deux modes différents ayant été exécutés sur les mêmes nœuds, ou deux expériences à fréquences égales sur les deux mêmes nœuds, on se rend compte que c'est essentiellement le temps de communication entre les tâches qui a beaucoup varié. Cette variation parasite perturbe le résultat global, alors que le changement de mode DVFS impacte uniquement le temps de calcul et n'influence pas le temps passé en communication. Les valeurs d'écart-type élevées montrent la dispersion non négligeable des mesures relevées, dûes grandement à la variation de performance du réseau de l'environnement utilisé. Pour le calcul énergétique, les PDUs disponibles sur le site de Reims ont été interrogés une fois par seconde, mais leurs mesures ne sont actualisées que par intervalle de 3 secondes ce qui nuit également au calcul de l'énergie consommée.

6. Conclusion

Le but de cet article était tout d'abord de décrire tous les outils nécessaires dans un simulateur afin d'être en mesure de simuler la consommation énergétique d'une ou plusieurs machines durant une expérience, de décrire la méthodologie à appliquer afin de pouvoir comparer et évaluer une implémentation du DVFS dans CloudSim sur un exemple simple. La première partie de l'article montre le bon fonctionnement du DVFS dans le simulateur, et la deuxième partie met en évidence la difficulté d'obtenir des valeurs de consommation fiables sur une plate-forme réelle, notamment dûes aux mesures

de puissance peu précises mais aussi aux temps d'exécution assez variables en fonction des ressources utilisées. Les futures expériences se dérouleront dans un environnement plus isolé avec des systèmes de mesures de puissances cumulées plus précis. Les travaux futurs avec le simulateur CloudSim seront dédiés au développement d'algorithmes de placement et gestion de qualité de service.

Bibliographie

1. Alexandru (M.), Monteil (T.), Lorenz (P.), Coccetti (F.) et Aubert (H.). – Efficient large electromagnetic problem solving by hybrid tlm and modal approach on grid computing. *In : International Microwave Symposium, Montréal, Canada, 17-22 june 2012*.
2. Bak (S.), Krystek (M.), Kurowski (K.), Oleksiak (A.), Piatek (W.) et Waglarz (J.). – Gssim-a tool for distributed computing experiments. *Scientific Programming*, vol. 19, n4, 2011, pp. 231–251.
3. Buyya (R.) et Murshed (M.). – Gridsim : A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *concurrency and computation : practice and experiene (ccpe)*, vol. 14, n13, 2002, pp. 1175–1220.
4. Calheiros (R. N.), Ranjan (R.), Beloglazov (A.), De Rose (C. A. F.) et Buyya (R.). – Cloudsim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software : Practice and Experience*, vol. 41, n1, 2011, pp. 23–50.
5. Casanova (H.), Legrand (A.) et Quinson (M.). – Simgrid : A generic framework for large-scale distributed experiments. *In : Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on*, pp. 126–131.
6. Da Costa (G.) et Hlavacs (H.). – Methodology of Measurement for Energy Consumption of Applications (regular paper). *In : Energy Efficient Grids, Clouds and Clusters Workshop (co-located with Grid) (E2GC2), Brussels, 25/10/2010-29/10/2010*. p. (electronic medium). – IEEE.
7. Hoeffer (W.). – The transmission-line matrix method–theory and applications. *Microwave Theory and Techniques, IEEE Transactions on*, vol. 33, n10, oct 1985, pp. 882–893.
8. Information Sciences Institute (U. o. S. C.). – The network simulator ns2.
9. Isci (C.) et Martonosi (M.). – Runtime power monitoring in high-end processors : Methodology and empirical data. *In : Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*. pp. 93–. – Washington, DC, USA, 2003.
10. Joseph (R.) et Martonosi (M.). – Run-time power estimation in high performance microprocessors. *In : Proceedings of the 2001 international symposium on Low power electronics and design*. pp. 135–140. – New York, NY, USA, 2001.
11. Keir (C. C.), Clark (C.), Fraser (K.), H (S.), Hansen (J. G.), Jul (E.), Limpach (C.), Pratt (I.) et Warfield (A.). – Live migration of virtual machines. *In : In Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI*, pp. 273–286.
12. Kliazovich (D.), Bouvry (P.), Audzevich (Y.) et Khan (S.). – Greencloud : A packet-level simulator of energy-aware cloud computing data centers. *In : GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, pp. 1 –5.
13. Kolpe (T.), Zhai (A.) et Sapatnekar (S.). – Enabling improved power management in multicore processors through clustered dvfs. *In : Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pp. 1 –6.
14. Kurowski (K.), Nabrzyski (J.), Oleksiak (A.) et Weglarz (J.). – Grid scheduling simulations with gssim. *In : Parallel and Distributed Systems, 2007 International Conference on*, pp. 1–8.
15. Limited (E. O.). – Wireless electricity meter. – `http://www.plogginternational.com/products.shtml`.
16. Ostermann (S.), Plankensteiner (K.), Prodan (R.) et Fahringer (T.). – Groudsim : An event-based simulation framework for computational grids and clouds. *In : Euro-Par 2010 – Parallel Processing Workshops*. – Springer.
17. Patterson (M.), Tschudi (B.), Vangeet (O.), Cooley (J.) et Azevedo (D.). – *ERE : A Metric for Measuring the Benefit of Reuse Energy from a Data Center*. – Rapport technique nWhite Paper 29, The Green Grid, 2010.
18. Rivoire (S.), Ranganathan (P.) et Kozyrakis (C.). – A comparison of high-level full-system power models. *In : Proceedings of the 2008 conference on Power aware computing and systems*. pp. 3–3. – Berkeley, CA, USA, 2008.